

Operadores Aritméticos e Operadores Lógicos

OPERADORES ARITMÉTICOS E LÓGICOS

Esses operadores são empregados com muita frequência em programação. É com o seu uso e, muitas vezes da combinação de vários deles, que são feitas as tarefas mais comuns de processamento de dados.

Operadores Aritméticos

OPERADOR	OPERAÇÃO	TIPOS DOS OPERANDOS	TIPOS DO RESULTADOS
+	adição	inteiro	inteiro
		real	real
-	subtração	inteiro	inteiro
		real	real
*	multiplicação	inteiro	inteiro
		real	real
/	divisão	inteiro	real
		real	real
DIV	divisão inteira	inteiro	inteiro
MOD	resto da divisão inteira	inteiro	inteiro

Operadores Relacionais

OPERADOR	OPERAÇÃO	TIPO RESULTANTE
=	igual	lógico
<>	diferente	lógico
<	menor que	lógico
>	maior que	lógico
<=	menor ou igual	lógico
>=	maior ou igual	lógico
<=	está contido	lógico
>=	contém	lógico
in	pertence a	lógico

Operadores Lógicos

OPERADOR	OPERAÇÃO
NOT	NÃO (negação)
AND	E (conjunção lógica)
OR	OU (disjunção lógica)
XOR	OU exclusivo

Observação: NOT é um operador unário.

Prioridades de operadores

Operadores aritméticos:

Prioridade mais alta: * / DIV MOD

Prioridade mais baixa: + -

Observações:

1. Parênteses garantem maior prioridade

✚ Exemplo: para a expressão **3*(4+1)** o resultado é 15 e não 13, como resultaria em **3*4+1**.

2. Se houver uma sequência de operadores de igual prioridade, a execução será na ordem em que eles aparecerem nas operações.

✚ Exemplo: o resultado da expressão **7/(3*2)** será diferente de **7/3*2**. Já no caso de **5*4/2** e de **5*(4/2)**, o resultado será exatamente o mesmo. Na dúvida, recomenda-se empregar parênteses para se ter a certeza de que a ordem de execução desejada seja cumprida.

Operadores lógicos:

Prioridade mais alta: NOT

Prioridade média: AND

Prioridade mais baixa: OR XOR

Exemplos:

✚ A expressão **(A>3) AND NOT (A>25)** equivale a **(A>3) AND (NOT (A>25))**

✚ Na expressão **(X=0) OR (X>=2) AND (X<=5)**, será resolvido primeiro o **AND**. Assim, a expressão equivale a **(X=0) OR ((X>=2) AND (X<=5))**

Observação: operações com mesma precedência normalmente são executadas da esquerda para a direita, embora o compilador possa, às vezes, rearranjar os operadores para otimização de código.

Importante: na dúvida, é aconselhável usar parênteses para garantir a prioridade desejada

Exemplo. Escrever um programa em Português Estruturado que, dados dois números inteiros e positivos, calcule e mostre o resto inteiro da divisão do primeiro número pelo segundo número.

```
Programa DIV_e_MOD
Var
N1, N2, R: inteiro;
Inicio
  Escreva ("Digite o primeiro número inteiro e positivo");
  LEIA (N1);
  Escreva ("Digite o segundo número inteiro e positivo");
  LEIA (N2);
  R ← N1 MOD N2;
  ESCREVA ("O resto inteiro da divisão do primeiro número pelo segundo número é", R);
FIM.
```

Desvios Condicionais – Comandos de Seleção

Desvios Condicionais (Comandos de Seleção)

Além dos algoritmos lineares, também chamados de algoritmos seqüenciais, há situações que demandam a introdução de comandos de seleção (desvios condicionais simples e compostos) para que decisões possam ser tomadas.

Condição

Em Lógica de Programação, **condição** é uma expressão lógica em função da qual uma decisão possa ser tomada. **Vejamos um exemplo:** imagine que um aluno somente é aprovado se sua média final, armazenada na variável identificada por MF, for igual ou superior a sete (7). A condição relacionada com esse exemplo é $MF \geq 7$.

Decisão

Decisão é a escolha de qual trecho do algoritmo deve ser executado de acordo com o resultado obtido e comparado com uma condição estabelecida.

No caso do exemplo anterior, se a média final do aluno for 8,5, o algoritmo poderia tomar a decisão de imprimir a frase "O aluno foi aprovado".

Como a condição na qual se baseia a decisão só admite duas respostas possíveis, a decisão envolverá, no máximo, duas alternativas: uma ação caso a condição seja verdadeira (V) e outra ação caso a condição seja falsa (F). Existe a possibilidade de haver apenas a primeira alternativa: nesse caso, se a condição for falsa (F), o algoritmo apenas seguirá o fluxo normal posterior ao desvio condicional, classificado como desvio condicional simples.

Observação: muitas vezes, indicamos o caso da condição verdadeira como "Sim" (S) e o caso de condição falsa como "Não" (N).

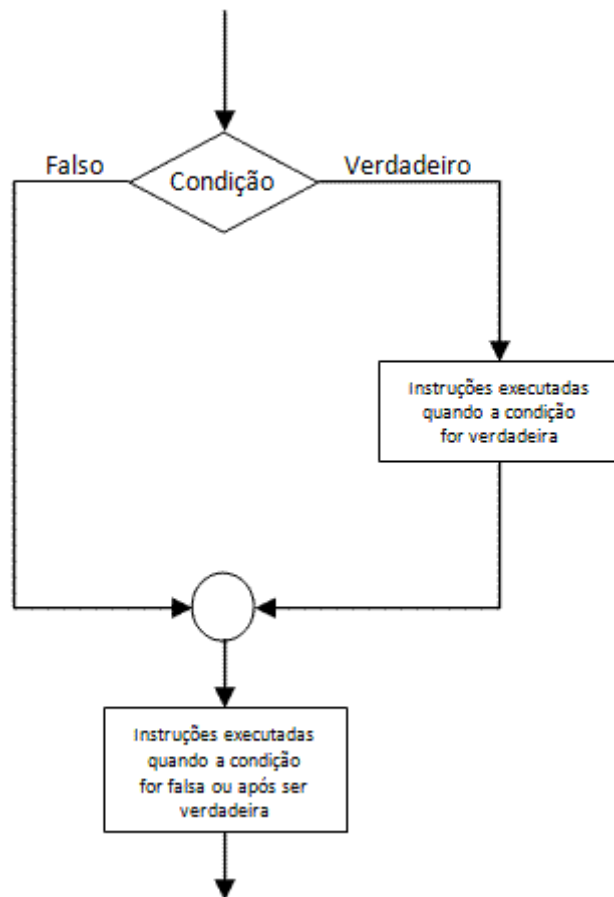
Desvio Condicional Simples

O desvio condicional simples é uma estrutura na qual se indica, no algoritmo, uma tomada de decisão em que um trecho do algoritmo é executado caso uma determinada condição seja verdadeira (V).

Em Português Estruturado, o desvio condicional simples é indicado com as seguintes instruções:

se ... então ... fim_se.

O fluxograma relacionado com o desvio condicional simples está ilustrado na figura abaixo.

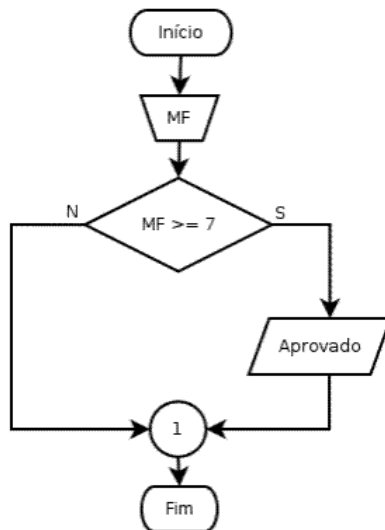


Suponha o seguinte problema: dada a média final de um aluno, indicada pela variável identificada por MF, mostrar apresentar a expressão "Aprovado" se esta média for igual ou superior a 7.

Um algoritmo simplificado para esse problema poderia ser:

- ✚ Ler a média final de um aluno e implicar esta média na variável MF.
- ✚ Se MF for igual ou superior a 7 (sete), mostrar "Aprovado".

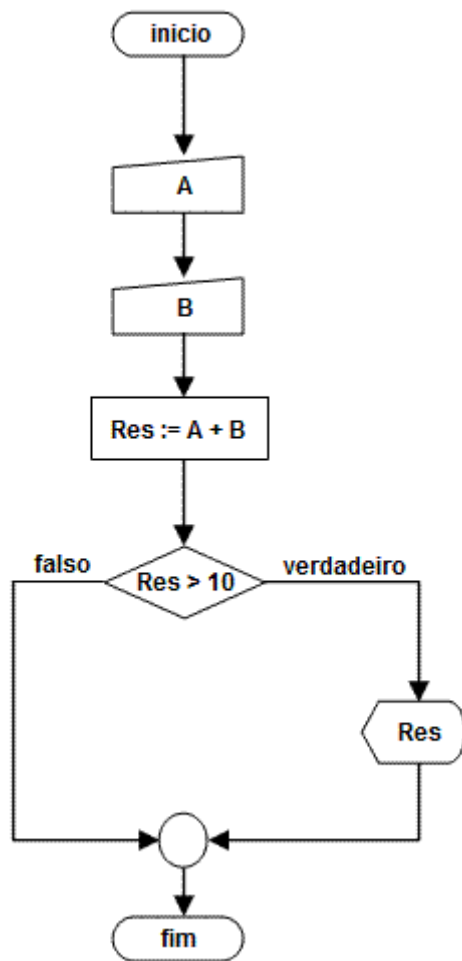
O fluxograma para o caso acima poderia ser o mostrado na figura que segue.



Em Português Estruturado:

```
Programa Media_Aluno
Var
  MF: real;
inicio
  leia (MF);
  se (MF>=7) então
    escreva ("Aprovado");
  fim_se;
fim.
```

Outro exemplo: fluxograma e programa em Português Estruturado envolvendo estrutura condicional simples.



```
Programa Soma_Resultado
var
  A, B, Res: inteiro;
inicio
  leia (A);
  leia (B);
  Res ← A + B;
  se (Res > 10) entao
    escreva (Res);
  fim_se;
fim.
```

Observação: na atribuição "Res recebe A + B", poderíamos ter feito $\text{Res} \leftarrow A + B$ ou $\text{Res} := A + B$ ou $\text{Res} = A + B$.

Desvio Condicional Composto

O desvio condicional composto é uma estrutura na qual se indica, no algoritmo, uma tomada de decisão em que um trecho do algoritmo é executado caso uma determinada condição seja verdadeira (V) e outra uma tomada de decisão em que um trecho do algoritmo é executado caso uma determinada condição seja falsa (F).

Desvio condicional composto

Sintaxe – Português Estruturado

```
se (condição) entao
    <instruções para condição verdadeira>
senão
    <instruções para condição falsa>
fim_se;
```

Exemplo. Ler dois valores numéricos e fazer sua adição. Caso o valor seja maior ou igual a 10, ele deverá ser apresentado somando-se a ele mais 5. Caso o valor somado não seja maior ou igual a 10, deverá ser apresentado subtraindo-se 7.

Programa em Português Estruturado

```
Programa Exemplo_Soma
var
    A, B, res : inteiro;
inicio
    leia (A);
    leia (B);
    res ← A + B
    se (res >= 10) entao
        res ← res +5;
    senao
        res ← res +7;
    fim_se;
    escreva ("O resultado é:", res);
fim.
```

Laços de Repetição, Contadores e Acumuladores

Laços ou Loops, Contadores e Acumuladores

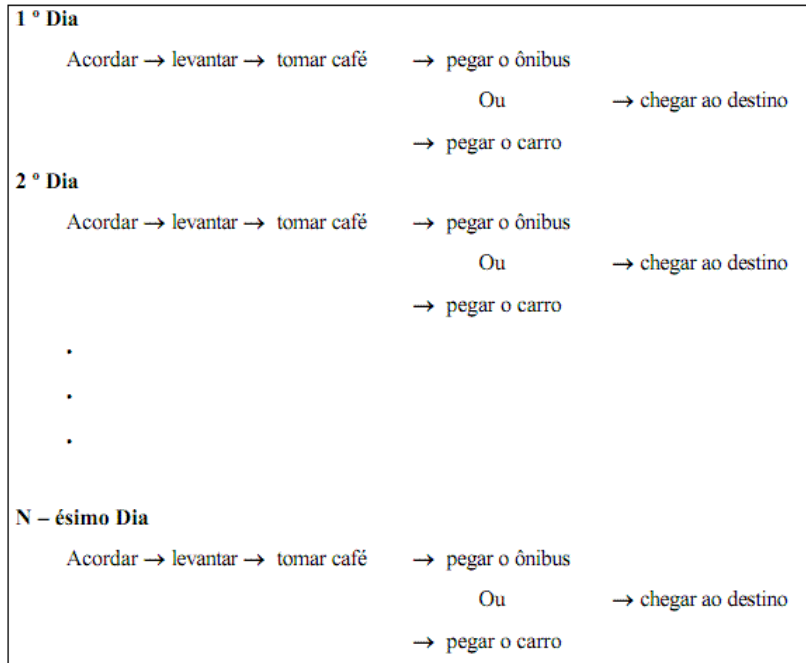
Laços ou Loops



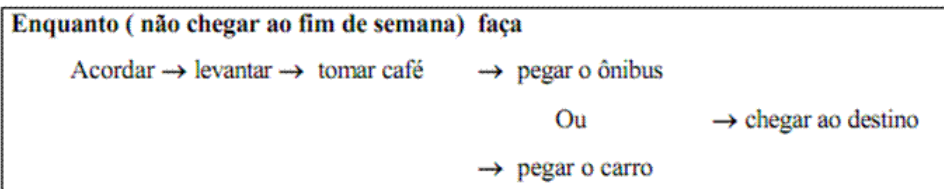
Muitas vezes, ao elaborarmos um algoritmo ou um programa de computador, precisamos repetir uma sequência de operações ou de instruções mais de uma vez. A maneira como essas repetições podem ser estruturadas são relacionadas com laços de repetição ou loops.

Uma sequência de ações é repetida por um número específico de vezes, até que uma condição seja satisfeita. Enquanto a condição for verdadeira (V), as instruções serão executadas.

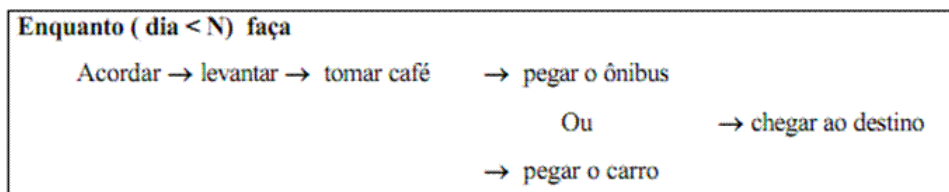
Exemplo. Rotineiramente, durante os dias da semana, uma pessoa realiza a sequência de ações mostradas no quadro abaixo.



Essa pessoa executa, repetidamente, ações descritas acima de segunda-feira a sexta-feira. Ou seja, as ações se repetem até que chegue o sábado. Sendo assim, podemos reescrever o quadro acima de uma maneira mais sintética, como mostrado nos quadros que seguem.



Ou



Concluimos que os esquemas apresentados nos últimos dois quadros representam, de modo bem mais "resumido", as N repetições de ações apresentadas no primeiro quadro. Pensando em termos práticos da programação, os modelos adotados nos dois últimos quadros são bem mais confortáveis e sintéticos do que o modelo do primeiro quadro.

Conforme veremos no próximo conteúdo, as estruturas de repetição são representadas por três tipos de comandos:

- ✚ repetição com teste no início,
- ✚ repetição com teste no final
- ✚ repetição com variável de controle.

Esses comandos podem realizar a repetição de ações (conjunto de comandos). Essas ações são executadas repetidamente enquanto uma determinada condição for satisfeita (verdadeira).

Variáveis – Contadores e Acumuladores

Em algoritmos contendo estruturas de repetição, é comum surgir a necessidade de utilizarmos variáveis do tipo contador e/ou acumulador.

Contadores

Um **contador** é utilizado para contar o número de vezes que um evento ou uma instrução ocorre, ou seja, contar a quantidade de vezes que uma instrução é executada.

De modo geral, um contador pode ser expresso como: $VARIÁVEL \leftarrow VARIÁVEL + CONSTANTE$

Podemos ler a expressão acima como "o novo valor da variável é o valor antigo da variável somado de um valor constante".

Vejamos um **exemplo**: $CONT \leftarrow CONT + 1$

Vejamos: um contador é uma variável qualquer, identificada por um símbolo escolhido pelo programador, que recebe "ela mesma mais um valor (uma constante)". No caso do exemplo $CONT \leftarrow CONT + 1$, a variável **CONT** está recebendo o valor dela mesma mais 1.

Usualmente, a constante que será somada no contador é o valor 1, para contar de "um em um", mas isso não é obrigatório. Poderíamos ter qualquer valor como constante, por exemplo o número 3, se desejássemos contar de 3 em 3.

Importante - Contadores:

- ✚ A variável do contador deve possuir um valor inicial conhecido, isto é, ela deve ser inicializada.
- ✚ Normalmente, inicializa-se a variável do contador com o valor zero, ou seja, zera-se a variável antes de utilizá-la. Para zerar uma variável basta atribuir a ela o valor zero: $VARIÁVEL \leftarrow 0$.
- ✚ A constante em $VARIÁVEL \leftarrow VARIÁVEL + CONSTANTE$, que geralmente é o valor 1, determina o valor do incremento da variável do contador, ou seja, o que será somado (acrescido) a ela.

Acumuladores

Um **acumulador** (ou somador) é utilizado para obter somatórios (representados pelo símbolo Σ).

De modo geral, um acumulador pode ser expresso como: $VARIÁVEL1 \leftarrow VARIÁVEL1 + VARIÁVEL2$

Podemos ler a expressão acima como "o novo valor da variável 1 é o valor antigo da variável 1 somado do valor da variável 2".

Vejamos um **exemplo**: $AC1 \leftarrow AC1 + AC2$

Vejamos: um acumulador ou somador é uma variável qualquer que recebe ela mesma mais uma outra variável. No caso de $AC1 \leftarrow AC1 + AC2$, a variável AC1 está recebendo o valor dela mesma mais o valor da variável AC2. A variável AC2 representa o valor a ser somado, acumulado na variável AC1.

Importante - Acumuladores:

- ✚ A variável 1 do acumulador deve possuir um valor inicial conhecido, isto é, ela deve ser inicializada. Normalmente, inicializa-se a variável do acumulador com zero, ou seja, zera-se a variável antes de utilizá-la. Para zerar uma variável, basta atribuir a ela o valor zero: $VARIÁVEL1 \leftarrow 0$.
- ✚ A variável 2 indica o valor a ser acumulado, somado e armazenado na variável 1.

Introdução

Há três estruturas básicas para a construção de algoritmos: os algoritmos sequenciais, os algoritmos com seleção e os algoritmos com repetição. A combinação dessas três estruturas permite a construção de algoritmos para a resolução de problemas complexos.

Uma estrutura de repetição permite que uma sequência de instruções ou comandos seja executada várias vezes, até que uma condição seja satisfeita, ou seja, repete-se um conjunto de instruções sem que seja necessário escrevê-las várias vezes. As estruturas de repetição, assim como a de decisão ou seleção, envolvem a avaliação de uma condição ou teste.

Estruturas de Repetição

As estruturas de repetição podem ser classificadas em *condicionais* e em *incondicionais*. Para todos os casos estudados a seguir, vamos tomar como exemplo uma situação hipotética de uma pesquisa de intenção de votos que será finalizada no momento em que o total de 20 (vinte) pessoas forem entrevistadas.

Estruturas de Repetição Condicionais

Os comandos colocados dentro de uma estrutura de **repetição condicional** têm sua execução repetida até que a **condição lógica** estabelecida seja satisfeita.

Temos, basicamente, dois tipos de repetições condicionais: o comando "**ENQUANTO...FAÇA**" e o comando "**REPITA ATÉ**".

ENQUANTO...FAÇA

Sintaxe Geral

```
ENQUANTO (condição) FAÇA
    bloco de comandos
FIM ENQUANTO
```

Exemplo

```
ENQUANTO (numero_pessoas <= 20) FAÇA  
    RECEBA intencao_votos  
FIM_ENQUANTO;
```

REPITA...ATÉ

Sintaxe Geral

```
REPITA  
    bloco de comandos  
ATÉ (condição)
```

Exemplo

```
REPITA  
    RECEBA intencao_votos  
ATÉ (numero_pessoas > 20)
```

A diferença básica entre essas duas estruturas é que a estrutura ENQUANTO...FAÇA testa a condição antes de executar o bloco de comando e a estrutura REPITA...ATÉ executa o bloco de comandos e depois testa a condição. Dessa maneira, a estrutura REPITA...ATÉ deve ser usada sempre que se queira executar o bloco de comandos pelo menos uma vez.

Observação: os operadores das condições dessas duas estruturas são inversos.

Estrutura de Repetição Incondicional

Os comandos colocados dentro de uma **repetição incondicional** têm sua execução repetida um determinado número de vezes.

PARA variável DE x ATÉ y PASSO z FAÇA

Esse comando pode ser utilizado tanto em sentido crescente (x menor do que y e z positivo), quanto em sentido decrescente (x maior do que y e z negativo).

Nele, temos o seguinte:

- ✚ x é o número inicial,
- ✚ y é o número final e
- ✚ z é o incremento/decremento que ocorrerá em *variável* em cada repetição.

Sintaxe Geral

```
PARA variável DE x ATÉ y PASSO z FAÇA  
    bloco de comandos  
FIM PARA;
```

Exemplo

```
PARA numero_pessoas DE 1 ATÉ 21 PASSO 1 FAÇA  
    RECEBA intencao_votos  
FIM PARA;
```

Observação. Repetição com Teste no Início: neste tipo de repetição há uma condição logo no início, e, enquanto esta condição for satisfeita (verdadeira) o conjunto de ações limitadas pelo bloco da repetição é executado. Quando a condição for falsa o comando é abandonado prosseguindo para o próximo comando na sequência.

